N89-15593

KNOWLEDGE REPRESENTATION ISSUES FOR EXPLAINING PLANS

Mary Ellen Prince
James D. Johannes

University of Alabama in Huntsville
Computer Science Department
Huntsville, AL 35899

ABSTRACT

Explanations are recognized as an important facet of intelligent behavior. Unfortunately, expert systems are currently limited in their ability to provide useful, intelligent justifications of their results. We are currently investigating the issues involved in providing explanation facilities for expert planning systems. This investigation addresses three issues: knowledge content, knowledge representation, and explanation structure.

Introduction

An important characteristic of an intelligent system, whether human or computer, is the ability to explain or justify its actions. Recognizing this fact, expert system developers were the first to regularly incorporate explanation facilities into their programs. [3]. Unfortunately, early attempts at automated explanation produced results that were significantly different from human explanations, both in organization and in information content.

In large measure this was due to the methods which were used to model expert problem solving knowledge. Production rules, the most popular form of knowledge representation, proved to be an effective representation for generating solutions, but were less than satisfactory for generating explanations of those solutions. The domain principles and expertise which determined the organization and content of the rule base were represented implicitly, if at all, and therefore could not be used to justify the system's behavior.

Recent research has focussed on methods of improving the explanation capability of intelligent systems. ([4], [6], [9], [10], [11]). At least three issues must be considered: first, identification of the kinds of knowledge which constitute a useful explanation; second, determination of a representation formalism to make the knowledge readily accessible to the explanation generator; and third, methods for selecting and organizing the knowledge in order to present it in a meaningful format.

This paper will address the first issue in depth and will suggest an approach to the second. For a discussion of explanation organization, see [2]. The explanation domain will be planning systems.

The Importance of Explanations

There is a significant gap between what users would like to see in an explanation system and what is feasible in light of current theory and technology. Although researchers are working to close the gap, much remains to be done.

A system that is able to explain its own behavior has several advantages over systems that lack this ability. For example, adequate explanation facilities can reassure skeptics that the system's reasoning processes are sound and its results are reliable. They can also serve a tutorial purpose. A properly constructed description of the strategy and domain principles used to derive a solution can provide insight which the user can then apply to other, similar, problems. The novice is thus encouraged to expand his knowledge of the domain, much as if he were working directly with a human expert. Explanations can be useful to knowledge engineers during the test and debug phase of system development, just as program traces are useful to a programmer under similar conditions ([1], [6]). In addition, good explanations can provide an automatic documenting capability. Finally, it can be argued that a system which contains the knowledge needed to produce good explanations can also be designed to use this information to improve its own performance in areas such as error recovery.

Current State of Explanation Technology

Explanation technology is severely limited in its ability to provide the benefits cited above. Explanations typically assume one of two forms: natural language traces of the rules currently under consideration or, less frequently, "canned text" inserted by the designer.

Rule traces have some advantages. They provide an accurate record of the program's activity and are thus helpful for debugging the knowledge base and for showing when the program is being pushed beyond the limits of its ability. In addition, modifications to the rules are automatically reflected in the associated explanations, thereby insuring consistency. Explanations produced by paraphrasing rules suffer, however, from several defects. They are poorly structured, and often filled with low level operational details that are of little interest to the average user. More seriously, they are in general incapable of explaining causal relations, the rationale behind a solution, strategic issues, or anything about fundamental domain principles. The reason for this is that the knowledge required for deep explanations is not explicitly represented in the rules. Problem solving strategy and domain relations are implicit in the clause ordering of rule concepts, and the principles which justify the rules and strategy are missing altogether.

Canned text explanations can be used to annotate individual rules or groups of rules, but this approach lacks the flexibility

which is needed for a full scale explanation system. It is difficult to anticipate every explanation which may be required. In addition, there is no guarantee that changes to the program code will be reflected in changes to the associated explanations, since there is no automatic connection between the two.

In summary, it is apparent that the current state of explanation technology falls far short of that which may be desired. The following sections will examine the kinds of questions that a good explanation system might reasonably be expected to address, and identify the knowledge needed to respond to these questions. For background, a discussion of the planning domain will be presented.

The Planning Domain

This paper addresses explanation generation in the context of expert planning systems. Previous explanation research has concentrated on diagnostic expert systems, primarily in the areas of medicine and electronic trouble shooting. An important consideration is whether these findings can be extended to other domains, such as planning. A comparison between planning systems and diagnostic systems will help to answer this question.

Simply stated, an expert planner is a system that generates a sequence of steps which, when applied from a given starting state, will produce the desired goal state. Some systems are interactive, so that feedback during plan execution can influence future planning decisions. In traditional planners, often called strategic planners, it is more commonly the case that plan generation and plan execution are two distinct processes. Robot problem solvers and automatic programming are two common application domains for planning systems.

The individual steps in a plan are produced by plan operators, which describe the legal actions or events that can occur in the application domain. Operators are typically described by a set of preconditions, which determine when an operator can legally be applied, and a set of postconditions, which describe the operator's effect on the world state. Operator descriptions may also include additional information such as procedures for accomplishing desired effects, ordering constraints, and resource requirements.

The planning process consists of choosing appropriate operators and ordering them to achieve the goals which constitute the final state. Early planners were linear; that is, they developed a sequence of steps to achieve each individual goal in order. At every point in the planning process the plan was fully detailed, but complete only to that point. STRIPS [7] is the most familiar example of a linear planner.

Hierarchical planners, on the other hand, start with a high level representation of the entire plan and refine it through

several levels of abstraction to arrive at the final sequence of primitive operators. A feature frequently associated with hierarchical planners is partial ordering of actions. Non-hierarchical planners are often forced to make arbitrary ordering decisions in order to maintain the linear nature of the incomplete plan. Hierarchical planners postpone committment until it is clear that the committment will not have to be undone at some later stage. NOAH[8], NONLIN[12], and SIPE[13] are hierarchical planners.

A third paradigm is case-based or script-based planning. This approach relies on a library of existing skeletal plans which are adapted to new situations by various refinement and "debugging" techniques. HACKER and MOLGEN[5] are two examples of planners that fit this paradigm.

Comparison of Planning Systems and Diagnostic Systems

Comparison of planning systems and diagnostic systems suggests several parallels that can be exploited to transfer explanation theory from the diagnostic domain to the planning domain. Two that will be investigated here are the knowledge bases and the inferencing processes.

The knowledge of a typical diagnostic system is encoded in production rules, a simple, flexible formalism for representing expert reasoning. Rules consist of two parts: the premise, typically a conjunction of clauses, and the conclusion, or goal. The conclusion can be established by proving that the clauses in the premise are true. This may be done by gathering evidence directly or by proving other rules which have the same clauses as goals. Thus the knowledge base can be viewed as a hierarchical network with implicit links between goals and premises.

The operators in a planning system serve a function analagous to that of production rules in the sense that they contain the necessary problem solving knowledge. The analogy can be extended to structural aspects of the knowledge as well. An operator's preconditions (premises) must be satisfied in order to achieve the desired postconditions (goal). Satisfying preconditions can be accomplished by applying other operators with the appropriate postconditions, thus giving the set of defined operators a network structure.

Inference in expert systems is accomplished by rule chaining. Depending on its design, a system may chain backward from a suspected diagnosis to known evidence, or it may chain forward from the evidence to a diagnosis.

Planners, particularly those based on the hierarchical paradigm, may use problem reduction as an inference technique. The method is to first express a plan as a sequence of high level goals and then to refine each abstract goal into a set of more concrete subgoals. The refinement process can be repeated as often as is

necessary to produce the final sequence of primitive actions. As an alternate, a planner might employ means-ends analysis. This approach involves a comparison between the current state and the goal state. Wherever differences are detected, operators are selected to reduce the differences. This is also an iterative process.

Plan derivation and diagnosis differ in the details of how the appropriate operators or rules are selected but the effect of the selection process is similar in both cases. At any point during the inferencing process there exists a stack of goals, implicit or explicit, which must be realized. An achieved solution represents a path through the network of rules or operators. For planning systems the path is equivalent to the plan; for diagnostic systems it represents the chain of reasoning that led to a specific diagnosis.

The solution produced by a planning system is more complex than the solution produced by a diagnostic system. A plan is a structured entity consisting of an ordered sequence of steps, while a diagnosis consists of a single entity. In addition, many planners operate in dynamic, multi-agent domains. They must plan simultaneous actions, prevent harmful interactions between competing agents, and consider the effects of actions over which they have no direct control. It is reasonable to expect that this added complexity will cause a corresponding increase in the complexity of the associated explanation. The next section will expand on this premise through a discussion of the epistemological issues of explanation theory as applied to expert planners. It will first outline some of the issues that must be addressed by an explanation system and will then present a taxonomy of explanation related knowledge.

The Epistemology of Explanations

A good explanation facility should be flexible enough to meet the needs of domain experts, novice users, and system designers. The following items illustrate the kinds of questions that it might be expected to address.

- Domain Facts, Principles, and Terminology
  Terminology is important as a foundation for understanding higher level explanations. Principles describe the problem solving procedures which can be used to achieve a goal. In well defined domains most facts can be expressed as causal relations, while in less formalized domains, empirical associations and heuristics play an important role.
- Comparisons and Choices
  Explaining why one action is preferable to another is a difficult task. Such decisions may be predicated on an accumulation of prior evidence, or on anticipation of future effects. In general, this kind of explanation requires a knowledge of constraints, interactions between events, and ultimate goals.

- Justification
  Justifying a single step in a plan can be as simple at stating a causal relationship or as complicated as explaining a choice. Justifying an entire plan may require the system to identify strategies, constraints, priorities, resource restrictions, and temporal issues.
- Methodology
  Questions about methodology refer to the mechanics by which a particular solution was obtained.
- General Strategy
  In addition to specific methods, most problem solvers also rely on abstract principles and weak methods to guide the problem solving process.

No system has yet been able to respond to all of these issues. Expert systems have traditionally answered questions about methodology by paraphrasing a chain of executed rules. In planning systems, a similar effect can be achieved by showing how operators in a general procedure have been instantiated with case-specific data. MYCIN[3] had a limited ability to compare alternative drug therapies. In NEOMYCIN Clancey[4] and Hasling et al [6] extended the explanatory capabilities of MYCIN by incorporating meta-rules to provide information about strategy. Shulman and Hayes-Roth[9] designed an explanation module to provide justifications and feasibility evaluations for certain knowledge systems where the reasoning was controlled by a strategic plan. In general, however, most systems lack the deep knowledge required to provide a broad range of explanations.

The remainder of this section identifies the kinds of knowledge needed for plan explanation. This identification is based on previous explanation research from the diagnostic domain, as well as on the specific needs of the planning domain. For purposes of discussion, the knowledge will be classified as either meta-knowledge (knowledge about knowledge), domain knowledge, or case-specific knowledge.

- Meta-Knowledge

Meta-knowledge embodies knowledge about control strategy and problem solving techniques. While a specific method or strategy might be classified as domain knowledge, the guidelines used to choose between competing strategies fall into the category of meta-knowledge. Many of the so-called "weak methods" can also be categorized this way. Examples of meta-knowledge are "Look for common causes for a device malfunction before looking for unusual causes" or "Avoid ordering plan operators until there is a reason to do so."

It is not clear that there are principles which are applicable to every planning domain. For example, a linear planner might employ the principle "Order plan operators arbitrarily if no

information exists; modify later if necessary" instead of the "avoid ordering" principle cited earlier. It is clear, however, that every planning system operates on a set of general strategic principles which may, in fact, have wide application.

Some illustrations of these general strategies may be found in the literature. Swartout[10], for example, recommends the use of "tradeoffs" and "preferences", where tradeoffs indicate the pros and cons of selecting a particular goal-achieving strategy and preferences are used to prioritize goals. Planners in multi-agent domains that permit parallel actions have devised methods for resolving the conflicts that arise when actions in one branch of a plan interfere with actions in another branch[13]. System designers must identify the abstract principles that guide their own problem solving and incorporate them into the meta-level knowledge structure. In order to provide good explanations of general strategy and to justify final plans it is important that the information be represented explicitly.

- Domain Knowledge

Without domain knowledge, it is impossible to explain terminology, principles, and domain facts. It is also difficult to furnish justifications and explanations of general prínciples unless domain specific information is available. Both declarative and procedural knowledge are required here. Declarative knowledge encompasses terminology and factual information, while procedural knowledge expresses how goals can be accomplished. Swartout and Smoliar [11] discuss the need for terminological, domain descriptive, and problem solving knowledge in the context of EES, an expert system which diagnoses cardiac difficulties and prescribes digitalis therapy. Their structure is sufficiently general to apply to planning as well as diagnostic domains.

The terminology of a planning system includes all domain concepts. Physical objects, their properties, and relations among objects such as "on-top-of" or "greater-than" must be defined in terms of system primitives. Factual knowledge can be represented as assertions of causal relations or probabilistic associations. Certain types of constraints which control the temporal ordering of operators and specify harmful or helpful interactions may also be represented this way.

Procedural or strategic knowledge in intelligent planners involves the selection and ordering of plan operators. The application of domain strategies is subject to control by meta-level knowledge and is, at the same time, dependent on case-specific information that can activate constraints and ordering rules. To be fully explainable, strategies must also be supported by a rationale based on domain facts.

- Case-Specific Knowledge

Every instance of a planner's operation begins with a speci-

fication of the initial world state, the desired goal state, and a list of constraints, availabe resources, and other pertinent information. Using meta-level and domain strategies, the planner then generates a sequence of steps which describe how to achieve the goal. The final plan consists of these steps, instantiated to satisfy the initial specifications.

While the plan itself may be used to explain methodology, much as a traditional diagnostic system uses its rule chain to explain its diagnosis, it is necessary to keep a case history of the problem-solving process in order to provide deep explanations. At a minimum, the case history must include the procedures used, choices made, and the reasons for those choices.

As has been previously noted, some choices occur when the planner is forced to decide among two or more operators. Other decisions determine the ordering of plan steps. Diagnostic systems use certainty factors or other numerical weights as an aid when making similar decisions. Quantitative values do not contain enough information to generate satisfactory explanations, however, nor are they always appropriate in the planning domain. Planning decisions result from a combination of constraints, goal priorities, resource availability, or the knowledge that one or more of the options would interfere with the achievement of some future goal. This is the type of knowledge that must be kept in the case history.

It should be clear from the preceding discussion that there is no absolute boundary separating meta-knowledge, domain knowledge, and case-specific knowledge. Furthermore, there are situations where it is necessary to integrate information from more than one knowledge level to produce an adequate explanation. The next section will investigate methods of structuring planning knowledge to make it accessible to the explanation generator.

Representation Issues

The knowledge required to explain plans is, on the whole, the same knowledge that is required to generate the plans. Previous intelligent systems have made much of this knowledge unavailable for explanation generation. The problem now is to develop representation formalisms that will make the information explicit without unduly affecting the efficiency of the plan generator. A completely developed representation scheme is beyond the scope of this paper. Instead, it will concentrate on outlining a general knowledge structure to guide future research.

Domain terminology is best represented as a type hierarchy of nodes. The highest levels of the hierarchy serve as an index to domain concepts, while the lowest level can be instantiated with case-specific data. Individual nodes have attributes which can be either pointers to other nodes, definitions, or other properties. The pointers define the hierarchy and permit property inheritance. Attributes describe concept features and may be used to record constraints on the values of plan variables.

Operators also have a natural hierarchical structure. Abstract operators encode meta-level strategies which in turn invoke domain procedures. At the bottom of the hierarchy are the primitive operators which define individual plan steps. In addition to parameters, pre-conditions, and post-conditions, operators should include information about constraints, resources, and rationales. Constraints may apply to variable values or to temporal ordering. Resource requirements describe the domain resources needed to perform the step and the duration for which the resources must be available. The rationale may state that the operator is necessary in order to establish some condition needed for a future action or it may provide a causal justification for the process invoked.

The case history records the refinement process by which the plan was generated, giving it, too, a hierarchical structure. The highest levels contain information about meta-level decisions, such as options between alternative strategies. Intermediate levels are concerned with domain dependent choices. The lowest level corresponds to the actual steps of the plan. Nodes in the history are instantiated with case-specific data, where appropriate. Choice nodes can be annotated with reasons that justify the choices. For explanation purposes it is vital that the domain facts and definitions that motivated the choices be represented. The structure will then contain all knowledge needed to justify the plan.

6. Conclusion

Explanation theory is just beginning to move beyond the narrow scope of early efforts. Providing intelligent responses to a variety of questions requires a full and explicit representation of the knowledge involved. The hierarchical nature of knowledge in an expert planning system enables the planning process to be explained on many levels of abstraction. Systems which rely primarily on the knowledge embedded in low-level rules forfeit this opportunity. Building the knowledge bases required for adequate explanations is no small task. It is an activity that requires careful attention from domain experts and knowledge engineers alike. The result of this effort, however, is a system that will be more responsive to the needs of its users.

REFERENCES

1. Berry, D. C. and Broadbent, D. E. "Expert Systems and the Man-Machine Interface", Expert Systems, vol. 4, no. 1, Feb. 1987, 18-28.

2. Bridges, S. and Johannes, J. D. "Explanation Production by Expert Planners", Proceedings of Fourth Conference on Artificial Intelligence for Space Applications, to be published. 1988.

3. Buchanan, B. G. and Shortliffe, E. H. Rule Based Expert Systems. Addison-Wesley, Reading, MA, 1984.

4. Clancey, W. J. "The Epistemology of a Rule-Based Expert System - A Framework for Explanation", Artificial Intelligence, 20, 1983, 215-251.

5. Cohen, P. and Feigenbaum, E. A. The Handbook of Artificial Intelligence, vol. 3. William Kaufman, Inc. Los Altos, CA, 1984.

6. Hasling, D. W., Clancey, W. J. and Rennels, G. "Strategic Explanations for a Diagnostic Consultation System". International Journal of Man-Machine Studies, 20, 1984, 3-19.

7. Nilsson, N. J. Principles of Artificial Intelligence. Tioga Publishing Co., Palo Alto, CA, 1980.

8. Sacerdoti, E. D. A Structure for Plans and Behavior. Elsevier North-Holland, Inc. New York, NY, 1977.

9. Schulman, R. and Hayes-Roth, B. ExAct: A Model for Explaining Actions. Knowledge Systems Laboratory, Report No. KSL 87-8, Stanford University, Stanford, CA, 1987.

10. Swartout, W. R. "Knowledge Needed for Expert System Explanation", Future Computing Systems, vol. 1, no. 2, 1986, 99-113.

11. Swartout, W. R. and Smoliar, S. W. "On Making Expert Systems More Like Experts", Expert Systems, vol. 4, no. 3, Aug. 1987, 196-207.

12. Tate, A. "Generating Project Networks", Proceedings IJCAI 77, 1987, 888-893.

13. Wilkins, D. E. "Domain-Independent Planning: Representation and Plan Generation", Artificial Intelligence, 22, 1984, 269-301.